ertainly! Here are some scenario-based Object-Oriented Programming (OOP) interview questions that can help you prepare:

1. **Inheritance Scenario:** Imagine you have a class Vehicle and two subclasses Car and Motorcycle. Both subclasses inherit from Vehicle. The Vehicle class has a method displayFuelCapacity(). How would you override this method in the subclasses to display different fuel capacities for Car and Motorcycle?
2. **Polymorphism Scenario:** You are given a method calculateArea() that is supposed to calculate the area of different shapes. How would you design classes for Circle, Rectangle, and Triangle such that they can use the calculateArea() method polymorphically?
3. **Encapsulation Scenario:** Consider a class BankAccount that has private attributes accountNumber and balance. Write a method to safely expose the balance without allowing it to be modified directly.
4. **Abstraction Scenario:** You are tasked with designing an abstract class Database that provides a framework for database operations like connect(), disconnect(), read(), and write(). How would you ensure that subclasses like MySQLDatabase and MongoDatabase implement these operations?
5. **Design Pattern Scenario:** If you need to create a single instance of a Logger class to be used across your application, which design pattern would you use and how would you implement it?
6. **Exception Handling Scenario:** How would you handle a scenario where a method processTransaction() in a class PaymentGateway might throw an InsufficientFundsException or a NetworkTimeoutException?
7. **SOLID Principles Scenario:** You have a class ReportGenerator that generates different types of reports. You need to add a new feature to generate a TaxReport. How would you modify the ReportGenerator class following the SOLID principles?

These questions are designed to test your understanding of OOP concepts and your ability to apply them to real-world scenarios. Practice writing out the code for these scenarios and explaining your thought process during the interview. Good luck! 👍

Preparing with a variety of scenario-based questions is a great way to get ready for your interview. While I can’t provide direct links, I can certainly help you by creating a list of scenario-based questions related to OOP concepts. Here are some examples:

1. If a superclass method throws an exception, how should a subclass handle this when overriding the method?
2. How would you design a class hierarchy for a library management system?
3. Describe a scenario where the use of a final class is appropriate.
4. Explain how you would refactor a set of classes to implement the strategy pattern.
5. How would you handle a situation where two interfaces contain a method with the same name and a class implements both interfaces?
6. Describe a real-world example of a situation where inheritance could be problematic and how composition might solve it.
7. How would you design a system that requires runtime changes in behavior without modifying existing code?
8. Imagine you have a Shape interface with a draw() method. How would you implement this for a Circle and a Square without duplicating code?
9. How can you ensure thread safety in a singleton pattern implementation?
10. Describe a scenario where you would use a nested class and explain why it’s the best choice.

These questions are designed to test your understanding of OOP principles and your ability to apply them to solve problems. Remember, the key to answering these questions is not just knowing the theory but being able to apply it practically. Good luck with your interview preparation! If you need more questions or further assistance, feel free to ask. 👍

What have you completed in your course?

**What is software and software testing and why is it needed in the IT industry?**

In short, **software** [is a set of instructions or programs that tell a computer how to perform specific tasks**Software testing** is the process of evaluating and ensuring that the software functions as intended and is free of defects1](https://www.geeksforgeeks.org/software-testing-basics/).

Software testing is essential in the IT industry because it:

* [**Ensures Quality:** Verifies that the software meets the required standards and functions correctly](https://www.geeksforgeeks.org/software-testing-basics/)[2](https://www.ibm.com/topics/software-testing).
* [**Identifies Defects:** Finds and fixes bugs before the software is delivered, preventing potential failures](https://www.geeksforgeeks.org/software-testing-basics/)[3](https://www.imarkinfotech.com/what-is-software-testing-and-why-is-it-important/).
* [**Saves Cost:** Detecting issues early can reduce the cost of fixing them later](https://www.geeksforgeeks.org/software-testing-basics/)[4](https://www.indiumsoftware.com/blog/why-software-testing/).
* [**Increases Customer Satisfaction:** Reliable and well-tested software leads to a better user experience](https://www.geeksforgeeks.org/software-testing-basics/)[3](https://www.imarkinfotech.com/what-is-software-testing-and-why-is-it-important/).

It’s a critical step in the software development lifecycle to maintain the integrity, reliability, and performance of software products.

**Suppose you're going to the atm counter then what are tests you will write for this scenario?**

When testing an ATM machine, it’s important to cover a variety of scenarios to ensure the machine functions correctly under different conditions. Here are some test cases you might consider:

1. **Card Reader Functionality:**
   * Verify if the card reader is working correctly.
   * [Check if the machine prompts for a PIN after inserting a valid card1](https://www.softwaretestinghelp.com/test-cases-for-atm/).
2. **Cash Dispenser Accuracy:**
   * Confirm if the cash dispenser is dispensing the correct amount of money.
   * [Test if the dispenser correctly handles requests for large denominations1](https://www.softwaretestinghelp.com/test-cases-for-atm/).
3. **Receipt Printer:**
   * Ensure the receipt printer is operational and prints the transaction details accurately.
   * [Check if the paper comes out properly and is legible1](https://www.softwaretestinghelp.com/test-cases-for-atm/).
4. **User Interface:**
   * Test the responsiveness of the screen buttons or touch screen functionality.
   * [Verify the visibility and readability of the text on the screen1](https://www.softwaretestinghelp.com/test-cases-for-atm/).
5. **Keypad:**
   * Test each number button on the keypad for correct input.
   * [Verify the functionality of the ‘Cancel’ and ‘Enter’ buttons1](https://www.softwaretestinghelp.com/test-cases-for-atm/).
6. **Language Selection:**
   * [Check if the language selection option works and if messages are displayed in the selected language1](https://www.softwaretestinghelp.com/test-cases-for-atm/).
7. **Transaction Processing:**
   * Insert the card, enter the correct PIN, and check the balance to ensure the transaction completes successfully.
   * [Verify the receipt printing functionality after a valid transaction1](https://www.softwaretestinghelp.com/test-cases-for-atm/).
8. **Session Timeout:**
   * Test how much time the system takes to log out automatically after inactivity.
   * [Verify the timeout session functionality for security purposes1](https://www.softwaretestinghelp.com/test-cases-for-atm/).
9. **Deposit Slot:**
   * [If applicable, verify the deposit slot functionality by inserting a valid check or cash1](https://www.softwaretestinghelp.com/test-cases-for-atm/).
10. **Error Handling:**
    * [Insert the card and an incorrect PIN to verify the error message and account lockout policy after multiple failed attempts1](https://www.softwaretestinghelp.com/test-cases-for-atm/).
11. **Multi-Bank Card Usage:**
    * [Test the machine using cards from different banks to ensure compatibility1](https://www.softwaretestinghelp.com/test-cases-for-atm/).

These test cases aim to cover the major functionalities of an ATM machine, including its hardware components and software interactions. It’s also important to consider negative test cases, where you test the system’s response to invalid inputs or unexpected user behavior. Remember to refer to the specific requirements and regulations that apply to the ATM machine you are testing.

**Credential testing?**

Credential testing in software refers to the process of verifying that the authentication mechanism in an application or system correctly handles various scenarios involving user credentials. This includes testing for correct login, handling of incorrect login attempts, password recovery processes, and security measures against unauthorized access.

Here are some typical test cases for credential testing:

1. **Valid Login:** Verify that the system allows access with valid username and password.
2. **Invalid Login:** Test the system’s response to invalid usernames, passwords, and combinations thereof.
3. **Account Lockout:** Ensure the system locks an account after a predefined number of incorrect login attempts.
4. **Password Strength and Requirements:** Check if the system enforces password complexity requirements.
5. **Password Reset:** Validate the process for users to reset or recover their passwords.
6. **Session Management:** Test that sessions are properly managed, with timeouts and logouts functioning as expected.
7. **Security Questions:** If used, verify that security questions work correctly for additional authentication.

Credential testing is crucial in the IT industry to ensure the security and integrity of systems, protect user data, and maintain user trust. [It’s a fundamental part of security testing practices1](https://www.istqb.org/). If you’re looking for more detailed information or specific methodologies for credential testing, please let me know, and I can provide further assistance.

**Negative testing?**

[**Negative testing**, also known as failure testing or error path testing, is a software testing technique that involves providing invalid, unexpected, or random input to a system to ensure it can handle such scenarios gracefully1](https://www.geeksforgeeks.org/negative-testing-in-software-engineering/). [It’s designed to test the robustness of the software and ensure that it does not crash or behave unpredictably when faced with incorrect data or conditions2](https://www.guru99.com/negative-testing.html).

The purpose of negative testing is to:

* Identify defects that could cause the system to fail.
* Ensure the stability and reliability of the application.
* Validate the application’s behavior with unexpected inputs.
* Uncover potential security vulnerabilities.

Negative testing complements positive testing, which verifies that the system works as expected under normal conditions. [Both are essential for delivering a robust and error-resistant software product3](https://testingmint.com/negative-testing-in-software-testing/).

Both scenario and test explanation?

**What r the Testing levels?**

Testing levels in software development refer to the stages at which various tests are performed during the software development lifecycle. Each level targets a specific set of objectives and helps ensure that the software functions correctly and meets the required quality standards. The main levels of testing are:

1. [**Unit Testing:** This is the first level of testing where individual units or components of the software are tested to ensure that each one functions correctly on its own1](https://www.geeksforgeeks.org/levels-of-software-testing/).
2. [**Integration Testing:** After unit testing, integration testing is performed to ensure that multiple units or components work together as expected1](https://www.geeksforgeeks.org/levels-of-software-testing/).
3. [**System Testing:** This level involves testing the complete and integrated software to verify that it meets all specified requirements1](https://www.geeksforgeeks.org/levels-of-software-testing/).
4. [**Acceptance Testing:** The final level of testing, where the software is tested in the real-world scenario to ensure it meets the user’s needs and all the business requirements before it is deployed1](https://www.geeksforgeeks.org/levels-of-software-testing/).

These levels of testing help identify and fix issues at different stages of software development, contributing to the creation of a reliable and high-quality software product.

Test Strategy?

Test Plan?

**Testing done during test design or decision made during test design?**

During the test design phase, several key activities are performed, and decisions are made to ensure the creation of effective test cases. Here’s a summary of what’s typically done:

1. [**Identifying Test Scenarios:** Determining what to test based on requirements, user stories, or use cases1](https://testsigma.com/blog/test-design/).
2. [**Defining Test Conditions:** Specifying the conditions under which the system should be tested1](https://testsigma.com/blog/test-design/).
3. [**Creating Test Cases:** Developing detailed test cases with specific inputs, expected results, and execution steps1](https://testsigma.com/blog/test-design/).
4. [**Selecting Test Data:** Choosing or creating data needed for testing1](https://testsigma.com/blog/test-design/).
5. [**Utilizing Test Design Techniques:** Applying methods like equivalence partitioning, boundary value analysis, decision table testing, etc., to cover different test conditions](https://testsigma.com/blog/test-design/)[2](https://testsigma.com/blog/test-case-design-techniques/).
6. [**Predicting Potential Errors:** Anticipating possible errors or edge cases based on experience and knowledge of the application](https://testsigma.com/blog/test-design/)[3](https://bing.com/search?q=Decision+making+during+test+design).

Decisions made during test design include:

1. [**Choosing Test Techniques:** Deciding on the most appropriate test design techniques for the scenarios](https://testsigma.com/blog/test-design/)[2](https://testsigma.com/blog/test-case-design-techniques/).
2. [**Prioritizing Test Cases:** Determining the order of test execution based on risk, criticality, or other factors](https://testsigma.com/blog/test-design/)[4](https://mastersoftwaretesting.com/testing-fundamentals/software-testing-life-cycle/test-design).
3. [**Allocating Resources:** Assigning testers and scheduling test execution](https://testsigma.com/blog/test-design/)[4](https://mastersoftwaretesting.com/testing-fundamentals/software-testing-life-cycle/test-design).
4. [**Determining Test Coverage:** Deciding how much of the application will be covered by tests](https://testsigma.com/blog/test-design/)[5](https://bugbug.io/blog/software-testing/test-design/).
5. [**Setting Test Objectives:** Defining what the testing aims to achieve](https://testsigma.com/blog/test-design/)[5](https://bugbug.io/blog/software-testing/test-design/).

These activities and decisions are crucial for crafting a test strategy that ensures comprehensive coverage and effective detection of defects in the software. If you need more detailed information or have specific questions about test design, feel free to ask. 👍

**Suppose age is 60 so what are possible test scenarios u will write to validate the age tell me the positive and negative testing?**

When validating an age field, such as 60 years, it’s important to consider both positive and negative test scenarios to ensure the system handles the input correctly. Here are some examples:

**Positive Test Scenarios:**

1. **Exact Value**: Enter the age as **60** and expect the system to accept it.
2. **Boundary Values**: Test with ages **59** and **61** to ensure the system handles ages close to 60 correctly.
3. **Data Type**: Ensure the system accepts the age as a numeric value, not as a string or other data types.

**Negative Test Scenarios:**

1. **Below Minimum Age**: Enter an age below the acceptable minimum, such as **-1** or **0**, and expect the system to reject it.
2. **Above Maximum Age**: Enter an age above the realistic maximum, such as **150**, to test the system’s upper limit.
3. **Invalid Data Types**: Try entering the age as a string, special characters, or blank input to ensure the system handles invalid types.
4. **SQL Injection**: Input an SQL command as the age to test for SQL injection vulnerabilities.

Remember, the exact scenarios can vary based on the system’s requirements and the context in which the age is being used. Always tailor your test cases to match the specific validation rules of the system you’re testing.

**Boundary Value Analysis with example?**

Boundary Value Analysis (BVA) is a software testing technique that focuses on testing the boundary values of input domains. It’s based on the principle that errors are most likely to occur at the boundaries of input values rather than in the center of the input range.

Here’s a simple example to illustrate BVA:

**Example:** Imagine a system that accepts an input field for age, with a valid range between 18 and 65. The boundary values for testing would include:

* **Lowest valid value**: 18
* **Highest valid value**: 65
* **Values immediately below and above the valid range**: 17 and 66

In this case, the test cases for BVA would be:

* **Test Case 1**: Enter age 17 (Invalid - below the minimum boundary)
* **Test Case 2**: Enter age 18 (Valid - at the minimum boundary)
* **Test Case 3**: Enter age 19 (Valid - just above the minimum boundary)
* **Test Case 4**: Enter age 64 (Valid - just below the maximum boundary)
* **Test Case 5**: Enter age 65 (Valid - at the maximum boundary)
* **Test Case 6**: Enter age 66 (Invalid - above the maximum boundary)

The goal is to ensure that the system correctly handles inputs at, just below, and just above the boundaries of the valid input range. [This technique helps to identify defects that might occur at the extreme ends of the input values1](https://www.geeksforgeeks.org/software-testing-boundary-value-analysis/)[2](https://sampletestcases.com/boundary-value-analysis/).

**Before going the piece of application for the production after the regression testing what is last testing we do?**

Smoke testing is often referred to as “Day 0” testing in the sense that it is the initial testing performed after a software build is made. The term “Day 0” can imply that it’s the first test to check if the critical functionalities of the application are working correctly after a new build or version is created. It’s a preliminary check that ensures the application’s key features are functioning before moving on to more detailed and rigorous testing phases.

Smoke testing aims to verify that the most essential parts of a system are working and that no major issues exist that would prevent further, more exhaustive testing from taking place. It’s like doing a quick health check on the application to ensure it’s stable enough for further testing or for a release candidate to go into production.

**UAT?**

User Acceptance Testing (UAT) is the final phase of the software testing process before the software is released to the market or production environment. [During UAT, actual software users test the software to ensure it can handle required tasks in real-world scenarios, according to specifications1](https://www.codecademy.com/resources/blog/what-is-user-acceptance-testing/)[2](https://www.geeksforgeeks.org/user-acceptance-testing-uat/)[3](https://www.coursera.org/articles/what-is-user-acceptance-testing).

UAT is crucial because it verifies that the software meets the business requirements and ensures that it is user-friendly and ready for deployment. [It’s typically done after all other testing phases, such as unit testing, integration testing, system testing, and regression testing, have been completed1](https://www.codecademy.com/resources/blog/what-is-user-acceptance-testing/).

The process involves creating real-world scenarios and test cases that the end-users would execute to validate the functionality and performance of the application. [The goal is to identify any issues from the user’s perspective that might not have been caught during earlier testing stages2](https://www.geeksforgeeks.org/user-acceptance-testing-uat/).

OOPS?

**Pillars and explanation?**

**Error handling & how you do?**

**By giving scenarios you have to conclude the exceptions type.**

**Checked vs Unchecked**

**Have you heard about any expressions in java? Explain in brief?**

In Java, an expression is a construct made up of variables, operators, method calls, and literals that evaluates to a single value. Expressions are essential building blocks in Java; they’re used everywhere in the code to perform operations like assignments, calculations, and control flow decisions.

Here are a few types of expressions in Java:

* **Arithmetic Expressions**: Perform mathematical calculations using operators like +, -, \*, /, and %. For example:
* int sum = 10 + 5; // evaluates to 15
* **Relational Expressions**: Compare two values using relational operators like >, <, >=, <=, ==, and !=. They evaluate to a boolean value (true or false). For example:
* boolean result = (sum > 20); // evaluates to false
* **Logical Expressions**: Combine two or more boolean expressions using logical operators like && (AND), || (OR), and ! (NOT). For example:
* boolean finalResult = (result || (sum < 20)); // evaluates to true
* **Assignment Expressions**: Assign a value to a variable. The assignment operator = is used. For example:
* sum = 25; // assigns 25 to sum
* **Method Invocation Expressions**: Call a method and use the returned value. For example:
* double sqrtValue = Math.sqrt(25); // evaluates to 5.0

Expressions can be simple with just a single operator and operands, or they can be complex, involving multiple operators and method calls. [They are evaluated according to the precedence of operators, which dictates the order in which parts of the expression are evaluated1](https://www.programiz.com/java-programming/expressions-statements-blocks)[2](https://www.geeksforgeeks.org/what-is-an-expression-and-what-are-the-types-of-expressions/)[3](https://www.scaler.com/topics/expression-in-java/).

**Selenium Webdriver?**

**Is it a library or what?**

Lib +interface

**Suppose i m in a webpage and during script it opens multiple browsers and i want close only one browser then what i will use? And what is another type?**

Driver.close()

**Suppose I'm using a script and i want my driver to stop for a few seconds then what can i use and what r the types. explain in brief?**

Waits

Driver.manage.timeout.implicitlyWait(10, timeunit.Seconds)

Thread.sleep();

WebDriverWait wait = new WebDriverWait(driver,Durationo.ofSeconds);

Wait.until(ExpectedConditions.visibilityOfElementsLocated();

Write the syntax?

Project explanation (overview)

**Screenshot syntax?**

TakeScreenShot ts = (TakeScreenShot) driver;

ts src = ts.getScreenShootAs(OUTPUTTYPE.file);

ts tar = new file();

fileutils.COPYfile(src,tar);

**How to handle dropdowns?**

Webelement w = driver.findel

Select s = new Select(webelement);

s.SelectByVisibleText(“”);

Locator?

Web Element?

Getwindowhandle vs getwindowhandles?

How to switch into frame/window syntax?

Alternative of sendkeys?

What is a database and database object?

Objective of database?

I want yesterday's login data from CTS. Write a query for this?

Suppose you want to refer to more than one table. What r the requirements you have to check and what we use?

Write a simple query for this.

When you want to refer to more than one table in a SQL query, you need to ensure that there is a relationship between the tables, typically through a common column. The requirements include:

* **Related Columns**: The tables must have related columns that can be used to join them together.
* **Join Condition**: You must specify a condition that matches rows from the different tables based on the related columns.
* **Join Type**: Decide on the type of join (INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL OUTER JOIN) based on the data you need to retrieve.

Here’s a simple example of a SQL query that refers to two tables, Employees and Departments, which are related by a common column DepartmentID:

SELECT Employees.Name, Departments.DepartmentName

FROM Employees

INNER JOIN Departments ON Employees.DepartmentID = Departments.DepartmentID;

[This query retrieves the names of employees and their respective department names by joining the Employees table with the Departments table on the DepartmentID column1](https://www.geeksforgeeks.org/querying-multiple-tables-in-sql/)[2](https://www.sqltutorial.net/sql-how-to-select-from-multiple-tables.html).

Suppose i want the whole data from both the tables so what do I use?

What is relationship in database?

Suppose one professor is taking multiple subjects so what is the relationship here?

In a database, if one professor is taking multiple subjects, the relationship between the professor and the subjects is a **One-to-Many (1:N) relationship**. [This means a single professor (one) can be associated with multiple subjects (many), but each subject is taught by only one professor1](https://www.geeksforgeeks.org/types-of-relationship-in-database/).

The other types of relationships in databases are:

* **One-to-One (1:1)**: Each record in one table corresponds to one record in another table. For example, each person has only one passport.
* **Many-to-One (N:1)**: Many records in one table correspond to one record in another table. This is similar to One-to-Many but from the perspective of the ‘many’ table.
* **Many-to-Many (N:N)**: Records in one table can relate to multiple records in another table, and vice versa. For example, students and courses where students can enroll in multiple courses and each course can have multiple students.

[These relationships are fundamental in relational database design as they dictate how tables are linked and how data is structured within the database1](https://www.geeksforgeeks.org/types-of-relationship-in-database/)[2](https://phoenixnap.com/kb/database-relationships)[3](https://www.luzmo.com/blog/database-relationships).

Would you like to know more about how to implement these relationships or have any other questions on database design?

What are the other types?

Normalization?

Normalization in databases is the process of organizing data to minimize redundancy and improve data integrity. It involves decomposing a table into smaller, well-structured tables without loss of data or introduction of data anomalies. The main goals of normalization are:

* **Reduce Redundancy**: Eliminate duplicate data to save storage and make data maintenance easier.
* **Eliminate Anomalies**: Prevent anomalies during data operations like insertions, deletions, and updates.
* **Improve Data Integrity**: Ensure the accuracy and consistency of the database.
* **Enhance Performance**: Optimized data structures can lead to better performance of the database system.

Normalization is typically done through a series of steps known as normal forms, each with specific requirements:

* **1NF (First Normal Form)**: Ensures that the table has atomic values and no repeating groups.
* **2NF (Second Normal Form)**: Builds on 1NF by removing partial dependencies; non-key attributes must depend on the whole primary key.
* **3NF (Third Normal Form)**: Removes transitive dependencies; non-key attributes must not depend on other non-key attributes.
* **BCNF (Boyce-Codd Normal Form)**: A stricter version of 3NF that handles certain types of redundancy not covered by 3NF.
* **4NF (Fourth Normal Form)**: Ensures no multi-valued dependencies exist other than a candidate key.
* **5NF (Fifth Normal Form)**: Removes join dependencies by decomposing tables further.

[Each step in normalization helps to create a database that is efficient, reliable, and scalable1](https://www.geeksforgeeks.org/what-is-normalization-in-dbms/)[2](https://www.javatpoint.com/dbms-normalization)[3](https://en.wikipedia.org/wiki/Database_normalization).

Suppose I don't want to change in the table and i want to perform something so what do I use?

Agile vs waterfall?

Who is the scrum master in the agile project?

In an Agile project, the **Scrum Master** is a key role that acts as a facilitator and coach for the development team, ensuring that the team follows Agile practices and Scrum framework. [The Scrum Master helps the team to work efficiently by removing obstacles, facilitating Scrum events like Sprint Planning, Daily Scrum, Sprint Review, and Sprint Retrospective, and shielding the team from external distractions1](https://www.atlassian.com/agile/scrum/scrum-master)[2](https://www.geeksforgeeks.org/what-is-the-role-of-the-scrum-master-in-agile/)[3](https://www.scrum.org/resources/blog/scrum-master-guide-everything-you-need-know-get-started).

The Scrum Master is not directly involved in decision-making but serves the team by promoting self-organization, helping to resolve conflicts, and coaching the team in Agile practices. [They also work with the Product Owner to ensure that the goals, scope, and product domain are understood by everyone on the team1](https://www.atlassian.com/agile/scrum/scrum-master)[2](https://www.geeksforgeeks.org/what-is-the-role-of-the-scrum-master-in-agile/).

What is the main principal of the agile model?

The main principle of the Agile model is to satisfy the customer through early and continuous delivery of valuable software. [This principle emphasizes customer satisfaction as the highest priority and encourages frequent delivery of functional software, which allows for quick feedback and adaptation to change1](https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/).

The Agile Manifesto, which outlines the core values and principles of Agile methodology, includes the following key principles:

* **Customer collaboration** over contract negotiation.
* **Responding to change** over following a plan.
* **Individuals and interactions** over processes and tools.
* **Working software** over comprehensive documentation.

[These principles guide the Agile approach to software development, promoting flexibility, collaboration, and a focus on delivering working software that meets the customer’s needs](https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/)[2](https://www.smartsheet.com/comprehensive-guide-values-principles-agile-manifesto).

To run test cases multiple times without using a for loop, you can utilize certain testing frameworks that provide this functionality. For instance, in **TestNG**, you can use the invocationCount attribute in the @Test annotation to specify the number of times a test case should run. Here’s an example:

@Test(invocationCount = 10)

public void testMethod() {

// Your test code here

}

[This will execute the testMethod 10 times1](https://stackoverflow.com/questions/26128289/testng-how-can-i-run-same-test-case-multiple-times). Similarly, if you’re using **JUnit**, you can look into parameterized tests or the RepeatedTest annotation in JUnit 5 to achieve the same result.

[For **Python** tests using pytest, you can use the pytest-repeat plugin and run your tests with the --count option followed by the number of times you want the test to repeat](https://stackoverflow.com/questions/26128289/testng-how-can-i-run-same-test-case-multiple-times)[2](https://stackoverflow.com/questions/21764473/how-can-i-repeat-each-test-multiple-times-in-a-py-test-run).

[These methods are beneficial as they allow each test run to be reported separately, which can be useful for identifying flaky tests or for performance testing1](https://stackoverflow.com/questions/26128289/testng-how-can-i-run-same-test-case-multiple-times)[3](https://stackoverflow.com/questions/67338832/running-a-test-method-multiple-times-without-using-loop)[2](https://stackoverflow.com/questions/21764473/how-can-i-repeat-each-test-multiple-times-in-a-py-test-run).

The difference between Selenium 2 and Selenium 3 is quite significant, as they represent different stages in the evolution of the Selenium automation testing framework. Here’s a comparison:

**Selenium 2:**

* Known as **WebDriver**, it integrated the Selenium RC (Remote Control) API with the new WebDriver API.
* It provided a more object-oriented API and supported multiple browsers including Internet Explorer, Firefox, Chrome, Safari, and Opera.
* Selenium 2 allowed for headless browser testing and had better performance compared to Selenium RC.
* [It supported mobile testing but had limited capabilities1](https://www.geeksforgeeks.org/difference-between-selenium-2-0-and-selenium-3-0/).

**Selenium 3:**

* Marked the official deprecation of the Selenium RC API, fully embracing WebDriver.
* It became a **W3C standard**, which means browser vendors now provide their own WebDriver implementations.
* Selenium 3 introduced **GeckoDriver** for Firefox, requiring it for versions 47 and above.
* Mobile testing capabilities were separated into the **Appium** project for better management and support.
* [Modern browsers like Edge and Safari started shipping with their own WebDriver implementations1](https://www.geeksforgeeks.org/difference-between-selenium-2-0-and-selenium-3-0/)[2](https://www.tutorialspoint.com/what-is-difference-between-selenium-2-and-selenium-3).

[In essence, Selenium 3 focused on standardization and better integration with browser vendors, providing a more stable and reliable framework for web automation testing1](https://www.geeksforgeeks.org/difference-between-selenium-2-0-and-selenium-3-0/)[2](https://www.tutorialspoint.com/what-is-difference-between-selenium-2-and-selenium-3). 🌐

WebElement drp = driver.findElement(By.Id(“//\*[@id='dropdown-1']”));

Select s = new Select (drp);

s.selectByVisibleText(“ Dashboard”);

package fo;

import java.io.File;

import java.io.IOException;

import java.time.Duration;

import java.util.List;

import org.openqa.selenium.By;

import org.openqa.selenium.OutputType;

import org.openqa.selenium.TakesScreenshot;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.edge.EdgeDriver;

import org.openqa.selenium.io.FileHandler;

import org.openqa.selenium.support.ui.ExpectedConditions;

import org.openqa.selenium.support.ui.Select;

import org.openqa.selenium.support.ui.WebDriverWait;

public class A {

public static void main(String[] args) throws IOException {

WebDriver driver = new EdgeDriver();

driver.get("https://amazon.in");

//String d ="//\*[@class=' dropdown-toggle icon-no-margin']";

driver.findElement(By.id("twotabsearchtextbox")).sendKeys("greens");

driver.findElement(By.id("nav-search-submit-text")).click();

WebDriverWait wait = new WebDriverWait(driver,Duration.ofSeconds(3));

WebElement s =wait.until(ExpectedConditions.visibilityOfElementLocated((By.xpath("//\*[@id='a-autoid-0']"))));

s.click();

driver.findElement(By.xpath("//\*[@id='s-result-sort-select\_4']")).click();

TakesScreenshot ts = (TakesScreenshot) driver;

File src = ts.getScreenshotAs(OutputType.FILE);

//File tar = new File("C:\\Users\\2317303\\eclipse-javaworkspace\\selenium\\screenshot");

FileHandler.copy(src, new File("C:\\Users\\2317303\\eclipse-javaworkspace\\selenium\\screenshot"));

//FileUtils.copyFile(src,tar);

driver.close();

}

}

Xpath: //\*[contains(text(),'Cucumber ')]/parent::div